Docket No. AUS920030429US1 PATENT APPLICATION

METHOD AND APPARATUS FOR PRELOADING TRANSLATION BUFFERS

BACKGROUND OF THE INVENTION

Field of the Invention

The invention relates generally to translations mechanisms in a computer architecture and, more particularly, to efficiently manage a translation mechanism to prevent problems associated with "warming" a translation cache.

Description of the Related Art

Many of today's processor architectures provide a translation mechanism for converting an effective address (EA) used by an application into a real address (RA) used for referencing real storage. One example of such a processor architecture is PowerPC™. The translation process uses a translation table to translate an EA to an RA. The translation table, or page table, is typically stored in memory. For performance reasons, a typical implementation of the translation mechanism uses a cache and/or buffering structure to hold recently used translations. This structure is referred to as a Translation Lookaside Buffer (TLB) in PowerPC™. Each instruction using an EA causes a lookup in the TLB. When a translation is not found in the TLB (for example, there is a

TLB demand miss), a hardware state machine or software routine is invoked to load the requested translation.

As with any caching mechanism, latency and bandwidth suffers when the cache does not contain a substantial amount of valid information required by an application. This condition is referred to as a "cold" cache. When a translation cache is cold, each access to a new area in storage causes a hardware or software action to be performed to load the requested translation. These demand misses continue until the translation caches are loaded with the most frequently used translations (for example, the translation cache is "warmed"). The additional latency and bandwidth degradation caused by the initial demand misses increase the runtime of an application. This condition typically occurs when a program is first run or when the processor swaps from one task to another, commonly referred to as the startup penalty. The startup penalty results in differences between the runtime of an application when executed on a "cold" versus a "warm" cache.

The startup penalty can be acceptable for non real-time applications. However, a real-time application should account for the worst-case latencies and bandwidth to guarantee a task can be completed in a specific amount of time (for example, a deadline). Therefore, real-time applications should account for the performance of a "cold" cache and, typically, cannot take full advantage of the

system performance. In addition, a real-time application that does not properly account for the performance differences between a "cold" and "warm" translation cache can miss a deadline.

Therefore, there is a need for a method and/or apparatus for avoiding the performance penalty of warming a cold cache that addresses at least some of the problems associated with the conventional demand miss methods and apparatuses for warming a cold translation cache.

SUMMARY OF THE INVENTION

The present invention provides a computer program product for managing a translation mechanism in a processor architecture, the computer program product having a medium with a computer program embodied thereon. A computer program code for transporting data through a data port is provided. Also a computer program code for supplying index data from an index table to the translator is provided. There is also a computer program code for providing management of the means for pre-loading.

BRIEF DESCRIPTION OF THE DRAWINGS

For a more complete understanding of the present invention and the advantages thereof, reference is now made to the following descriptions taken in conjunction with the accompanying drawings, in which:

FIGURE 1 is a block diagram depicting a conventional software-controlled translation mechanism;

FIGURE 2 is a block diagram depicting a conventional hardware-controlled translation mechanism;

FIGURE 3 is a block diagram depicting a Software-controlled Pre-load Translation Mechanism; and

FIGURE 4 is a block diagram depicting a Hardware-controlled Pre-load Translation Mechanism.

DETAILED DESCRIPTION

In the following discussion, numerous specific details are set forth to provide a thorough understanding of the present invention. However, those skilled in the art will appreciate that the present invention can be practiced without such specific details. In other instances, well-known elements have been illustrated in schematic or block diagram form in order not to obscure the present invention in unnecessary detail. Additionally, for the most part, details concerning network communications, electro-magnetic signaling techniques, and the like, have been omitted inasmuch as such details are not considered necessary to obtain a complete understanding of the present invention, and are considered to be within the understanding of persons of ordinary skill in the relevant art.

It is further noted that, unless indicated otherwise, all functions described herein can be performed in either

hardware or software, or some combinations thereof. In a preferred embodiment, however, the functions are performed by a processor such as a computer or an electronic data processor in accordance with code such as computer program code, software, and/or integrated circuits that are coded to perform such functions, unless indicated otherwise.

Referring to FIGURE 1 of the drawings, the reference numeral 100 generally designates a conventional software-controlled translation mechanism implementation. The Translation Mechanism Implementation 100 comprises Translation Mechanism 104 and a Software TLB Management Interface 102. The Translation Mechanism 104 comprises an Execution Unit (EU) 110, a Translation Lookaside Buffer (TLB) 112, a Software Miss Handler 114, and a Main Storage 116. The Main Storage 116 further includes a Page Table 118. In addition, Main Storage 116 can also include memory mapped I/O devices and registers. The Software TLB Management Interface 102 comprises a TLB Data Port 106 and a TLB Index 108.

Within the translation mechanism implementation 100, there is a plurality of interconnected devices that each perform specific tasks. The EU 110 executes instructions, such as instructions contained in an executable file. Instructions using an Effective Address (EA) to reference Main Storage 116 cause the EU 110 to forward the EA to the TLB 112 for translation. The TLB 112 searches the

translation buffer or cache for a translation for the EA. If there does not exist a translation for the EA issued by the EU 110, then the Software Miss Handler 114 searches for the unavailable, required translation in the Page Table 118 by computing the proper RA to locate the translation entry needed to translate EA provided by the EU 110 in the Page Table 118. The Software Miss Handler 114 is typically executed in the EU 110 or another processor in the system. Once the proper translation has been found for the EA 110 requested EA, the translation is loaded into the TLB 112, utilizing the Software Control Interface 102. The translation can now be used for future reference and the current EA is converted into a Real Address (RA) based on the data found in the Page Table 118. If the translation is not found in the Page Table 118, the Software Miss Handler 114 typically invokes a separate software mechanism (not shown) to resolve the translations missing in the Page Table 118. Missing translations result due to certain portions of the Page Table 118 being swapped to a mass media device such as a hard drive to more efficiently make use of processor memory, typically when translation entries in the swapped portion of the Page Table 118 have not been used in a lengthy period of time.

Within the Translation Mechanism 104, there exist a variety of connections to allow for the operation of the Mechanism 104 as described. The EU 110 is coupled to the

TLB 112 through a first communication channel 126, wherein the first communication channel 126 transfers an EA to the TLB 112. The TLB 112 is coupled to the Software TLB Management Interface 102 through a second communication channel 120 and a third communication channel 122. The second communication channel 120 and the third communication channel 122 each provide control data to the TLB 112. Also, the second communication channel 120 and the third communication channel 122 are used by the Software Miss Handler 114 to load translations found in the Page Table 118 into the TLB 112. The TLB 112 is further coupled to the Software Miss Handler 114 through a fourth communication channel 128, wherein a TLB Miss is communicated from the TLB 112 to the Software Miss Handler 114. TLB 112 is also coupled to the Main Storage 116 through a fifth communication channel 132, wherein an EU's 110 translated RA is communicated from the TLB 112 to the Main Storage 116. The Software Miss Handler 114 is coupled to the Page Table 118 through a sixth communication channel 130. The sixth communication channel 130 is used by the Software Miss Handler 114 to search the Page Table 118 for the translations missing in the TLB 112. Also, the EU 110 is coupled to the Main Storage 116 through a seventh communication channel 134, wherein data is intercommunicated between the EU and the Main Storage 116.

Within the Software TLB Management Interface 102, there exist a variety of connections to allow for the operation of the interface. The TLB Data Port 106 is coupled to the TLB 112 of the Translation Mechanism 104 through the second communication channel 120, wherein translation data is transferred from the TLB Data Port 106 to the TLB 112. The TLB Data Port 106 provides a communication port for delivering missing translations to the TLB 112. The TLB Index 108 is coupled to the TLB 112 of the Translation Mechanism through the third communication channel 122. Index data is communicated from the TLB Index 108 to the TLB 112 through the second communication channel 122. The TLB Index 108 contains the buffer location in the TLB 112 for the missing translations supplied by the TLB Data Port 106.

Now referring to FIGURE 2 of the drawings, the reference numeral 204 generally designates a conventional hardware-controlled Translation Mechanism Implementation. The Translation Mechanism Implementation 204 comprises an EU 210, a TLB 212, a Hardware Miss Handler 214, and a Main Storage 216. The Main Storage 216 further includes a Page Table 218. In addition, Main Storage 216 can also include memory mapped I/O devices and registers.

Within the Translation Mechanism Implementation 200, there is a plurality of interconnected devices that each performs specific tasks. The EU 210 executes instructions such as those contained in an executable file. Instructions

using an EA to reference Main Storage 216 cause the EU 210 to forward the EA to the TLB 212 for translation. The TLB 212 searches the translation buffers or cache for a translation for the EA. If there does not exist a translation for the EA issued by the EU 210, then the Hardware Miss Handler 214 searches for the unavailable, required translation in the Page Table 218. Once the proper translation has been found, the translation is loaded into the TLB 212 for future reference and the current EA is converted into an RA. The RA is then communicated to the Main Storage 216 through a fourth communication channel 232. Once the RA has been transmitted, data can be effectively transferred between the Main Storage 216 and the EU 210. If the translation is not found in the Page Table 218, the Hardware Miss Handler 214 typically invokes a software mechanism to resolve translations missing in the Page Table 218.

Within the Translation Mechanism 204, there exist a variety of connections to allow for the operation of the Mechanism 204. The EU 210 is coupled to the TLB 212 through a first communication channel 226, wherein the first communication channel 226 transfers an EA to the TLB 212. The TLB 212 is coupled to the Page Table 218 through a second communication channel 224, wherein the second communication channel 224 provides control data intercommunicated between the TLB 212 and the Page Table

218. The second communication channel 224 is used by the Hardware Miss Handler 214 to load translations found in the Page Table 218 into the TLB 212. The TLB 212 is further coupled to the Hardware Miss Handler 214 through a third communication channel 228, wherein a TLB MISS is communicated from the TLB 212 to the Hardware Miss Handler 214. TLB 212 is also coupled to the Main Storage 216 through the fourth communication channel 232, wherein an EU's 210 translated RA is communicated from the TLB 212 to the Main Storage 216. The Hardware Miss Handler 214 is coupled to the Page Table 218 through a fifth communication channel 230. The fifth communication channel 230 is used the Hardware Miss Handler 214 to search the Page Table 218 for the translations missing in the TLB 112. Also, the EU 210 is coupled to the Main Storage 216 through a sixth communication channel 234, wherein data is intercommunicated between the EU 210 and the Main Storage 216.

Referring to FIGURE 3 of the drawings, the reference numeral 300 generally designates a Software-controlled Pre-load Translation Mechanism. The Software-controlled Pre-Load Translation Mechanism 300 is similar to the Software-controlled Translation Mechanism Implementation 100 of FIGURE 1, with the inclusion of an additional Software Pre-Load Mechanism 301. The TLB Pre-Load Translation Mechanism 300 comprises a Software Pre-Load Mechanism 301, a Software-controlled Translation Mechanism 304, and a Software TLB

Management Interface 302. The configurations of Mechanism 304 and of Software TLB Management Interface 302 are substantially similar to the Mechanism 104 and Software TLB Management Interface 102 of FIGURE 1, respectively.

Within the Software TLB Management Interface 302, there exist a variety of connections to allow for the operation of the interface. The TLB Data Port 306 is coupled to the TLB (not shown but substantially similar to TLB 112 of FIGURE 1) of the Translation Mechanism 304 through the first communication channel 320, wherein translation data is transferred from the TLB Data Port 306 to the Translation Mechanism 304. Also, the TLB Index 308 is coupled to the Translation Mechanism 304 through a second communication channel 320. Index data is communicated from the TLB Index 308 to the Translation Mechanism 304 through the second communication channel 322. The TLB Index 308 contains the buffer location for the missing translations supplied by the TLB Data Port 306.

The Software Pre-load Mechanism 301 distinguishes the Software-controlled Pre-load Translation Mechanism 300 of FIGURE 3 from any other conventional Translation Mechanism Implementations, such as the Translation Mechanism Implementation 100 of FIGURE 1. The Software Pre-Load Mechanism 301 is coupled to the Software TLB Management Interface 302 through a third communication channel 311. The Software Pre-load Mechanism 301 with an extension of the

Software TLB Management Interface 302 allows translations to be pre-loaded into a TLB (not shown) from a Page Table (not shown) prior to the running of an application. In addition, the extensions allow for the state of the TLB (not shown) to be saved and restored when swapping tasks running on the execution unit. Pre-loading and restoring of the TLB provide for a reduction in the lag time by warming the associated TLB (not shown). Furthermore, the combination also allows for re-initializing the TLB when switching the context of the processor as opposed to a simple save and restore.

The Software Pre-load Mechanism 301 provides the applications with an interface for requesting the pre-load of translation. The requested translations can also be used to re-initialize the translations when switching the context of the processor. The interface can be an extension of the memory advise or "madvise" operating system call.

The "madvise" call includes an effective address and region size parameter which defines the start and size of an area in Main Storage for which translations are needed by an application. When receiving a "madvise" call, the Software Pre-load Mechanism 301 searches the Page Table (not shown) for the translations for the memory area defined by the parameters. Once the translations are found, the Software Pre-load Mechanism 301 loads the translation into the TLB (not shown) using the Software TLB Management Interface 302.

Referring to FIGURE 4 of the drawings, the reference numeral 400 generally designates a Hardware-controlled Pre-Load Translation Mechanism. The Hardware-controlled Pre-Load Translation Mechanism 400 is similar to the hardware-controlled Translation Mechanism Implementation 204 of FIGURE 2, with the inclusion of an additional Software Pre-Load Mechanism 401 and a Software TLB Management Interface 402.

The Hardware-controlled Translation Mechanism Implementations 400 is distinguished from any other conventional Hardware-controlled Translation Mechanism Implementations, such as the Implementation 200 of FIGURE 2. Included in the Implementation 400 are a Software TLB Management Interface 402 and a Software Pre-Load Mechanism 401. The Hardware-controlled Translation Mechanism Implementation 400 also comprises a Translation Mechanism 404. Moreover, the configuration of the Mechanism 404 is substantially similar to the Mechanism 204 of FIGURE 2.

The operation of the Software Pre-load Mechanism 401 in the Implementation 400 is similar to the operation of the Software Pre-Load Translation Mechanism 301 of FIGURE 3. However, to allow for the Software Pre-load Mechanism to work in a hardware-controlled mechanism, a Software TLB management interface is required. The interface is typically not included in conventional Hardware-controlled

mechanism since the TLB is managed by hardware miss handlers.

Within the Software TLB Management Interface 402, there exist a variety of connections to allow for the operation of the interface. The TLB Data Port 406 is coupled to the TLB 412 (not shown) of the Translation Mechanism 404 through the first communication channel 420, wherein translation data is transferred from the TLB Data Port 406 to the Translation Mechanism 404. The TLB Data Port 406 provides a communication port for delivering missing translations to the Translation Mechanism 404. The TLB Index 408 is coupled to the Translation Mechanism 404 through a second communication channel 422. Index data is communicated from the TLB Index 408 to the Translation Mechanism 404 through the second communication channel 422. The TLB Index 408 contains the buffer location for the missing translations supplied by the TLB Data Port 406.

Included with the Hardware-controlled Pre-Load Mechanism 400 is a Software Pre-Load Mechanism. The Software Pre-Load Mechanism 401 is coupled to the Software TLB Management Interface 402 through a third communication channel 411. The Software Pre-load Mechanism 401 with an extension of the Software TLB Management Interface 402 allows translations to be pre-loaded into a TLB (not shown) from a Page Table (not shown) prior to the running of an application. In addition, the extensions allow for the

state of the TLB (not shown) to be saved and restored when swapping task running the execution unit. Pre-loading and restoring of the TLB (not shown) provide for a reduction in the lag time by warming the associated TLB (not shown). Furthermore, the combination also allows for re-initializing the TLB when switching the context of the processor as opposed to a simple save and restore.

The Software Pre-load Mechanism 401 provides the applications with an interface for requesting the pre-load of translation. The requested translations can also be used to re-initialize the translations when switching the context of the processor. The interface can be an extension of the memory advise or "madvise" operating system call.

The "madvise" call includes an effective address and region size parameter which defines the start and size of an area in Main Storage for which translation are needed by an application. When receiving a "madvise" call, the Software Pre-load Mechanism 401 searches the Page Table (not shown) for the translations for the memory area defined by the parameters. Once the translations are found, the Software Pre-load Mechanism 401 loads the translation into the TLB (not shown) using the Software TLB Management Interface 402.

There are advantages and disadvantages to both a hardware and software-managed TLB (not shown). For example, the latency for resolving a TLB miss is less in a hardware-managed TLB mechanism than a software-managed TLB mechanism.

However, there is less control of the Page Table structure and the translations contained in the TLB of a hardware-controlled TLB mechanism. The Hardware-controlled Pre-load Translation Mechanism 400 of FIGURE 4 further includes a configurable Hardware Miss Handler (not shown), which invokes a Software Miss Handler (not shown) when the translation is not found in the TLB (not shown). The inclusion of a configurable Hardware Miss Handler (not shown) allows the system software to choose the best method for managing the translations required by an application.

From the foregoing description, it is understood that it is also possible to having varying degrees of concurrent control and management of a given Translation Lookaside Buffer. Hence, there are multiple embodiments of the present invention that can encompass varying degrees of control and/or management with respect to software and hardware.

It will further be understood from the foregoing description that various modifications and changes can be made in the preferred embodiment of the present invention without departing from its true spirit. This description is intended for purposes of illustration only and should not be construed in a limiting sense. The scope of this invention should be limited only by the language of the following claims.